(1)

PREPROCESSING SYSTEM FOR HARDWARE AUTOMATIC DESIGN SYSTEM

Patent Number: JP4023076
Publication date: 1992-01-27

Inventor(s): TOKUDA MASASHI
Applicant(s):: RICOH CO LTD

Requested Patent: JP4023076
Application Number: JP19900127164 19900517
Priority Number(s):
IPC Classification: G06F15/60
EC Classification:
Equivalents:

Abstract

PURPOSE:To efficiently perform the design work by unifying a language for verification and a language for function description.
CONSTITUTION:Function specifications used for verification of logic circuit functions are inputted to an input part 11 as they are, that is, function specifications where an algorithm is described in the C language (language for function description) are inputted there, and the same processing as a preprocessor in the C language processing system like file inclusion, character string substitution, macro substitution, or conditional compilation indicated by #inc/ude, #define, #ifdef, or the like is performed, and the results are transferred to an analyzing part 12. The analyzing part 12 performs deletion processing of parts unnecessary for logic design out of function specifications, memory information request, structure analysis, or the like while having conversations with a user through a user interface part 13, and the results are stored in an analysis result storage part 15. Thus, misconversion between two languages is eliminated to efficiently perform the design work.

(19)日本国特許庁(JP)　(11)特許出願公開

(12)公開特許公報(A)　平4-23076

(51)Int. Cl.[5]	識別記号	庁内整理番号	(43)公開　平成4年(1992)1月27日
G 06 F 15/60	360 K	7922-5L	

審査請求　未請求　請求項の数 1　(全5頁)

(54)発明の名称　ハードウエア自動設計システムの前処理方式

(21)特　願　平2-127164
(22)出　願　平2(1990)5月17日

(72)発明者　徳田　正志　東京都大田区中馬込1丁目3番6号　株式会社リコー内
(71)出願人　株式会社リコー　東京都大田区中馬込1丁目3番6号

明細書

1．発明の名称

ハードウエア自動設計システムの前処理方式

2．特許請求の範囲

1．論理回路の機能仕様を入力する入力部と，ユーザとの会話を制御するユーザインターフェース部と，前記論理回路の機能仕様を解析する解析部と，解析結果から論理合成システム用のCADデータを生成するCADデータ生成部とを備えて論理回路の機能仕様から論理回路を合成するハードウエア自動設計システムにおいて，前記入力部は論理回路機能を検証するときに用いた機能仕様をそのまま入力し，機能仕様の不要部分と使用メモリ情報の指示を前記ユーザインターフェース部で得て，変数の入出力と結合関係及び演算の発生順序等の解析を行ない，ハードウエア自動設計システム用のCADデータを生成するようにしたことを特徴とするハードウエア自動設計システムの前処理方式．

3．発明の詳細な説明

［産業上の利用分野］

本発明は，論理回路の機能仕様からハードウエアを設計するためのハードウエア自動設計システムの前処理方法に関するものである．

［従来の技術］

近年LSIの大規模化及びカスタム化に伴い，論理設計にコンピュータを用いて設計の効率化と共に大衆化も図られている．一般にこれらはハードウエア自動設計システムと呼ばれており，設計したいハードウエアの機能を記述して，それを論理回路として自動的に合成し，シミュレーションによって確かめられるようなシステムである．このようなシステムでは，ハードウエアの機能を特別な言語で記述する方法（以下，このような言語を機能記述用言語ということにする）が知られている．例えば，特開平2-39376号公報では論理合成の入力として特別なRTL言語を用意している．又，一般にこのような言語でハードウエアの機能記述を行う前に，アルゴリズムが正しいかどうかを検証しなければならないが，これには

汎用コンピュータ言語である高級プログラミング言語（以下、このような言語を検証用言語ということにする）を使ってプログラミングして確かめているのが現状である。

［発明が解決しようとする課題］

このように、考案したアルゴリズムをハードウエア化しようとするときには、通常高級プログラミング言語の検証用言語を使って正常に動くかどうかをコンピュータを使って検証し、アルゴリズムの確認を行っている。この確認作業の後、改めて機能記述用言語で記述し直してからハードウエア自動設計システムで論理回路を作成するので、

・２つの言語を学習しなければならない
・機能仕様を再度記述する
・書き直しでの変換ミスが入り込む

という問題があった。

本発明は、上記のような問題を解決するために、検証用言語と機能記述用言語とを一本化して、ハードウエア設計者が２つの言語を覚えなくても良いようにして、２言語間の変換ミスをなくし且つ設計作業全体での効率化を図ることを目的とする。

［課題を解決するための手段］

上記目的を達成するために請求項１に記載されている発明は、論理回路の機能仕様を入力する入力部と、ユーザとの会話を制御するユーザインターフェース部と、前記論理回路の機能仕様を解析する解析部と、解析結果から論理合成システム用のＣＡＤデータを生成するＣＡＤデータ生成部とを備えて、論理回路の機能仕様から論理回路を合成するハードウエア自動設計システムにおいて、前記入力部は論理回路機能を検証するときに用いた機能仕様をそのまま入力し、機能仕様の不要部分と使用メモリ情報の指示をユーザインターフェース部で得て、変数の入出力と結合関係及び演算の発生順序等の解析を行ない、ハードウエア自動設計システム用のＣＡＤデータを生成するようにしたことを特徴としている。

［作　用］

先ず、考案されたアルゴリズムは検証用言語で記述され、コンピュータによって検証される。

次に、その検証用言語で書かれたアルゴリズムはそのまま機能記述用言語で書かれたものとして入力され、不要部分やハードウエア化に必要な情報をユーザとの会話によって得てから、解析部で構造解析し、ハードウエア自動設計システムで使用できるようなＣＡＤデータを生成出力する。

［実　施　例］

以下に本発明の一実施例を図面を用いて詳細に説明する。本実施例では、検証用言語／機能記述用言語をＣ言語としているが、他の高級プログラミング言語であっても良いことはいうまでもない。

第１図は、本発明におけるハードウエア自動設計システムの実施例を示すブロック図である。１０は機能記述用言語で書かれた機能仕様を、１１は入力部を、１２は解析部を、１３はユーザインターフェース部を、１４はＣＡＤデータ生成部を、１５は解析結果記憶部を、１６はＣＡＤデータ記憶部を示している。

入力部１１に、アルゴリズムをＣ言語（機能記述用言語）で記述した機能仕様を入力し、#include, #define, #ifdef 等により指示されたファイルの包含、文字列の置換、マクロの置換及び条件付きコンパイル等、Ｃ言語処理系のプリプロセッサと同じ処理を行う。その結果は解析部１２へ渡される。

解析部１２は機能仕様のうち論理設計に不要な部分をユーザインターフェース部１３を通じてユーザと会話しながら削除処理、メモリ情報の要求及び構造解析等を行い、解析結果記憶部１５へ格納する。

第２図は本実施例における解析部の処理の流れを示すフローチャートである。

第３図に示したクイックソートのアルゴリズムを回路設計する例を、第２図を用いてより詳細に説明する。

ステップ１

ユーザインターフェース部１３は、ユーザにハードウエア化するのに関係のない部分を質問形式で入力を促す。この指示から不要部分

を削除すると、第４図のようになる。

ステップ２

更に、ユーザインタフェース部１３は、第５図のようにユーザとの会話によって、配列等のメモリアロケートで確保する記憶領域がどのようなメモリを使用するかの情報（種類、サイズ、構成、スピード等）を収集する。

ステップ３

ステップ１で不要部分を削除した機能仕様に対して、識別子、キーワード、定数、文字列、演算子、区切り等を判定し、入力を単語とその属性との対応表を作成するという、所謂字句解析を行う。

ステップ４

機能仕様と字句解析の結果とから、次のようなステップで構造解析を行う。

・関数単位に分割する。

・変数は定義された場所により次の２つのタイプに分類して抽出する。

関数外のとき：外部変数。

関数内のとき：内部変数。

演算子に使われていると、

それに関係する入力と出力の

変数名と演算子を一組とする。

・文の並びや制御文if-then, for, while, do-while, goto, 関数呼出し等の出現順序から、演算の発生順序を解析する。

（但し、関数の再帰的呼出しはスタックを用いて実現する。）

ステップ５

その解析結果を解析結果記憶部１５に格納する。

ＣＡＤデータ生成部１４は、解析結果記憶部１５にある解析結果をハードウエア自動設計システムへ入力する形式（例えば、第６図のようなデータフローとタイムチャート、及び第７図のようなメモリ情報等の情報）を生成し、ＣＡＤデータ記憶部１６へ格納する。

〔発明の効果〕

以上述べたように、請求項１に記載されている発明によれば、ハードウエア設計のための特別な言語を学習する必要がなくなり、設計に専念できる。又、アルゴリズムの検証用言語をそのまま入力するため変換ミスがなくなり、全体の設計時間も減少する。ハードウエア設計に詳しくないシステムエンジニアでもハードウエアの設計ができるようになる。更に、独自のハードウエア自動設計システムに本発明に基づく前処理方式を組み込むことにより汎用の高級プログラミング言語入力によるハードウエア自動設計システムが構築できる。

４．図面の簡単な説明

第１図は本発明の実施例を示すブロック図、第２図は解析部の処理の流れを示すフローチャート、第３図は検証用言語（機能記述用言語）で書かれた回路設計のアルゴリズムを示すプログラム例、第４図は不要部分を削除したプログラム例、第５図はユーザインターフェース部の会話例、第６図はデータフロー＆タイムチャートの生成例、第７図はメモリ情報の生成例である。

１０…機能仕様、１１…入力部、

１２…解析部、

１３…ユーザインターフェース部、

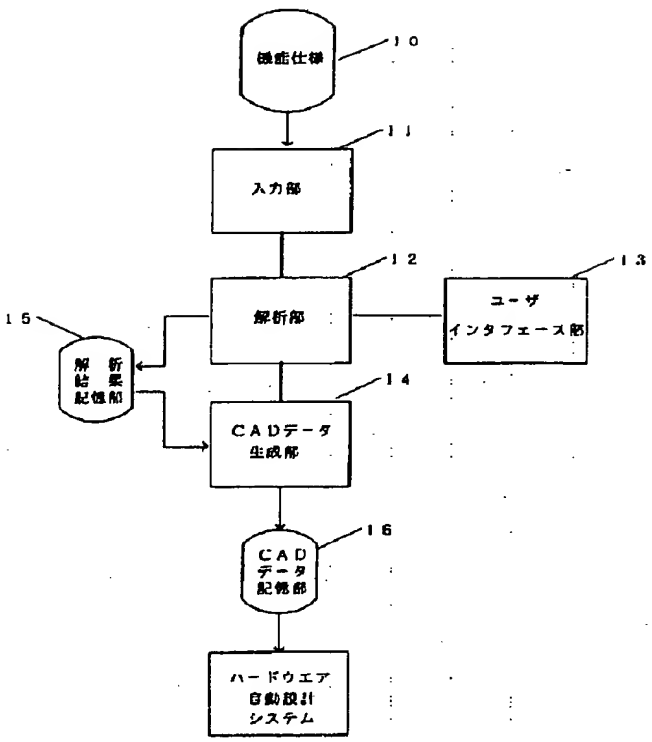
１４…ＣＡＤデータ生成部、

１５…解析結果記憶部、

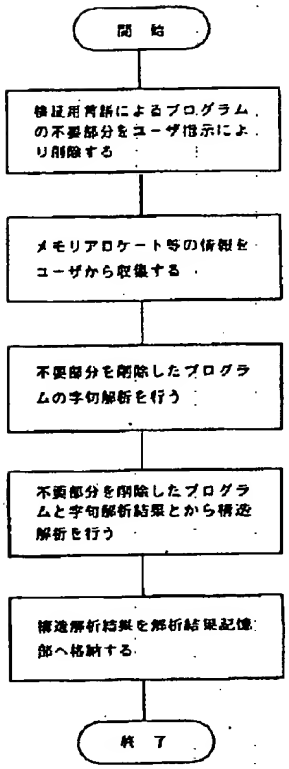
１６…ＣＡＤデータ記憶部。

特許出願人　　株式会社リコー

第1図



第2図



第3図

```
#include <stdio.h>
#define BUF_SIZE 12

int  *buf;
main()
{   int  i;

  buf = (int *)malloc(sizeof(int)*BUF_SIZE);
  if(buf == NULL) {
    printf(" Can't Allocation Memory\n");
    exit(0);
  }
  for(i=0; i<BUF_SIZE; i++)
    buf[i] = rand();
  quicksort(0, BUF_SIZE-1);
  for(i=0; i<BUF_SIZE; i++)
    printf(" %d", buf[i]);
  printf("\n");
}
quicksort(l, r)
int  l, r;
{   int  i = l, j = r;
    int  x, tmp;
  x = buf[(l + r)/2];
  do {
     while (buf[i] < x) ++i;
     while (x < buf[j]) --j;
     if(i <= j) {
       tmp = buf[i]; buf[i] = buf[j];
       buf[j] = tmp; ++i; --j;
     }
  } while (i <= j);
  if (l < j) quicksort(l, j);
  if (i < r) quicksort(i, r);
}
```

第4図

```
int  *buf;
main()
{   int  i;
  quicksort(0, size);
}
quicksort(l, r)
int  l, r;
{   int i = l, j = r;
    int  x, tmp;
  x = buf[(l + r)/2];
  do {
     while (buf[i] < x) ++i;
     while (x < buf[j]) --j;
     if(i <= j) {
       tmp = buf[i]; buf[i] = buf[j];
       buf[j] = tmp; ++i; --j;
     }
  } while (i <= j);
  if (l < j) quicksort(l, j);
  if (i < r) quicksort(i, r);
}
```

第５図

malloc関数の構成を定義してください。
１）外部メモリ
２）内部メモリ
のいずれですか？

[1]

メモリの属性は
１）ＳＲＡＭ
２）ＤＲＡＭ
３）ＲＯＭ
のいずれですか？

[1]

１ワードのビット巾は？

[1 6] ビット

メモリサイズは？

[4] ワード

アクセス時間は？

[1 0 0] ｎｓ

第７図

```
Name1 : MEM1
Type  : SRAM
Cycle : 100ns
Bits  : 8
Words : 128K
```

第６図

```
 0  func quicksort(l, r)   ; 関数の宣言
 1  l → i                  ; 変数の値の転送
 2  r → j
 3  l + r → buf          ; 加算して結果を転送
 4  buf / 2 → buf
 5  buf → ADDRESS      ; メモリから変数へ転送
    READ , MEM1, x
 6  i → ADDRESS
    READ , MEM1, buf2
 7  if (buf2 < x) then goto 8 else goto 9
 8  i+1 → i
    goto 6
 9  j → ADDRESS
    READ , MEM1, buf3
10 if (x < buf3) then goto 11 else goto 12
11  j-1 → j
    goto 9
12  if(i <= j) then goto 13 else goto 18
13  buf2 → tmp
14  buf3 → buf2
15  tmp → buf3
16  i+1 → i
17  j-1 → j
18  if (i <= j) then goto 5 else goto 19
19  if (l < j) then goto 21 else goto 20
20  if (i < r) then goto 23 else goto 25
21  Recursive  quicksort(l, j)
                ; スタックを利用した再帰呼出し
22  Recursive  quicksort(i, r)
```